UCRL-JRNL-225523

Fracture and Fragmentation of Simplicial Finite Elements Meshes using Graphs

A. Mota, J. Knap, M. Ortiz

October 24, 2006

International Journal for Numerical Methods in Engineering

Fracture and Fragmentation of Simplicial Finite Elements Meshes using Graphs

Alejandro Mota[1], Jaroslaw Knap[2] and Michael Ortiz[1] *

[1]Division of Engineering and Applied Science
California Institute of Technology
Pasadena, CA 91125, USA

[2]Chemistry and Materials Science Directorate
Lawrence Livermore National Laboratory
Livermore, CA 94550, USA

October 18, 2006

Abstract

An approach for the topological representation of simplicial finite element meshes as graphs is presented. It is shown that by using a graph, the topological changes induced by fracture reduce to a few, local kernel operations. The performance of the graph representation is demonstrated and analyzed, using as reference the 3D fracture algorithm by Pandolfi and Ortiz [22]. It is shown that the graph representation initializes in $O(N_E^{1.1})$ time and fractures in $O(N_I^{1.0})$ time, while the reference implementation requires $O(N_E^{2.1})$ time to initialize and $O(N_I^{1.9})$ time to fracture, where N_E is the number of elements in the mesh and N_I is the number of interfaces to fracture.

1 Introduction

The initiation and propagation of cracks and eventual fragmentation of a solid are among the most complex physical processes to which advanced computer simulations provide access. Within the field of computational solid mechanics, several different methods has been explored to simulate the fracture and fragmentation of solids: finite-element methods [1, 4, 14, 21], extended finite-element methods [9] and element-free Galerkin methods [5]. In conventional finite-element methods two distinct approaches to simulating fracture have become *de facto* standard: the smeared crack model and the discrete crack model (c.f. de Borst et al. [10] for a detailed review). In the smeared crack model, individual cracks are not explicitly resolved; instead, their effect on a solid is incorporated via deterioration of the material stiffness and strength.

By contrast, the discrete crack model is considerably more intuitive: a crack is introduced directly into the finite element discretization of the solid as a displacement discontinuity. This approach regards fracture as a *free-discontinuity problem*, i. e., a problem in which the displacement field may develop discontinuities

*Corresponding author. Email: ortiz@aero.caltech.edu

over a *singular* or *jump set* that is not known *a priori*. The mechanical behavior of the crack may conveniently be modeled by recourse to cohesive theories of fracture, and the evolution of the singular set may be achieved by the adaptive insertion procedure of surface-like cohesive elements at bulk-element interfaces [6, 21]. The insertion is customarily governed by a criterion evaluated at bulk-element interfaces throughout the course of a simulation. These adaptively-inserted cohesive elements capture the gradual de-cohesion process of the crack surfaces as the fracture proceeds.

Another class of problems that may be viewed and analyzed as free-discontinuity problems concern processes of strain and damage localization such as spallation and shear banding. For instance, solids deforming at high strain rates often develop narrow layers of high strain. Outstanding features of these strain localization zones are ultra-high local strain rates, local temperature raises, and high propagation speeds [16, 25, 28, 31, 34, 35]. Cracks, whether as a result of ductile fracture or of microvoid growth and coalescence often form along these shear bands [16, 32]. Moreover, spallation in metals may be regarded as the result of a process of damage localization leading to the formation of void sheets [8, 29]. These strain localization layers may be regarded strictly as a *sub-grid* phenomenon and, consequently, the bands of strain localization may be modeled as *displacement discontinuities*. These displacement discontinuities are confined to volume-element interfaces and are enabled by the insertion of specialized *strain-localization elements*. These elements consist of two surfaces, attached to the abutting volume elements, that can separate and slip relative to each other. The kinematics of the strain-localization elements is identical to the kinematics of cohesive elements proposed by Ortiz and Pandolfi [21] for the simulation of fracture [33].

In a simulation that uses cohesive elements or strain localization elements, the finite element mesh is initially coherent. As the simulation advances, a failure criterion is computed at bulk-element interfaces each time a specified number of computational steps have been performed. When the failure criterion is satisfied at a particular interface, a cohesive or localization element is introduced into the mesh. This element in turn governs the mechanics of the fracture or strain localization process at the interface. The continuous evolution of the topology of the mesh is an inherent feature of both cohesive element and localization element approaches. This evolution involves complex operations on the mesh, with the result that robust three-dimensional implementations that take into account these changes in topology are difficult to develop. A formal approach for the manipulation of the topology of finite element meshes is thus necessary in order to bring the complexities of these operations to a manageable level, and also to avoid the creation of *ad hoc* algorithms that may result in error-prone and inefficient implementations.

Mesh descriptions often assume one of the following two representations: *full* or *reduced*. According to the full representation, all topological entities in a mesh, such as points, segments or faces, are represented explicitly in the data structure. By contrast, in the reduced representation, one or more classes of topological entities are not represented explicitly, and hence their topological information must be inferred in terms of other entities that exist in the data structure [12]. The most common representation of a finite element mesh is a connectivity list or table. In this representation, each row in the table simply lists the nodes for a particular element according to some ordering convention. The connectivity table, although simple to implement, has proved extremely cumbersome for complex topological manipulations of meshes, such as those imposed by the aforementioned cohesive element and localization element methods.

Most advanced mesh representations are capable of effectively dealing with issues of non-manifold topologies, storage requirements, efficiency of construction, and modification or retrieval of topological information. For instance, the radial-edge data structure introduced by Weiler [30] emphasizes completeness and suitability for the representation of non-manifold topologies. Here, completeness is defined as the ability to generate all topological information from the representation alone, i.e., all adjacency relationships are directly retrievable or derivable from the information contained in the data structure [7, 24, 30]. Specialized

topology-based data structures were introduced by Beall and Shephard [3], with a varying degree of explicitness. Some of the proposed data structures are fully explicit, while the rest offer different levels of implicit representation with the objective of minimizing storage.

Previous efforts to develop a representation of a mesh, in which —possibly radical— topological changes occur, often involve the direct manipulation of finite-element connectivity arrays by means of complex algorithms [22, 23]. While this approach may be effective for relatively small meshes and light computational loads, it ostensibly fails to deal properly with non-manifold topologies and does not scale well to large meshes. Recently, Celes et al. [7] developed an implicit data structure that supports fracture and fragmentation. The authors particularly stress storage reduction and the representation of various types of elements, including simplicial ones. A direct comparison with our approach, however, is not possible at this time as the authors do not report performance data for fracture.

In the present work, a *complete* approach for the topological representation of simplicial finite element meshes as graphs is developed. By recourse to algebraic topology and graph theory, the original n-dimensional simplicial complex is reduced to a uni-dimensional simplicial complex in the form of a graph, thus greatly decreasing the complexity of topological manipulations. We show that the graph representation is particularly well-suited for the simulation of fracture and fragmentation. Some implementation issues are discussed, followed by a performance comparison of the graph representation with the reference algorithm of Pandolfi and Ortiz [22, 23], and the simulation of the brittle fracture of a gypsum cylinder subjected to a pressure pulse as a numerical example.

2 Finite Element Meshes as Graphs

We define a finite element mesh as a simplicial complex $K \in \mathbb{R}^N$ of dimension $n \leq N$ such that a simplex σ in the $(n-1)$-skeleton of K, $\sigma \in K^{(n-1)}$, is a face of at least one simplex of dimension n.

An oriented simplex $+\sigma$ is a simplex σ together with a particular ordering of its points and all even permutations thereof. The same simplex with all odd permutations of this ordering is said to have an opposite orientation and is denoted as $-\sigma$. A p-chain on K is a function $c : K \mapsto \mathbb{N}$ such that $c(\sigma) = -c(-\sigma)$ and $c(\sigma) = 0$ for all but finitely many oriented p-simplices in K [11, 19]. The elementary chain c corresponding to σ is defined as

$$c(\tau) := \begin{cases} 0, & \text{if } \tau \neq \sigma; \\ 1, & \text{if } \tau = \sigma; \\ -1, & \text{if } \tau = -\sigma. \end{cases} \tag{1}$$

The symbol σ is often used to denote both a simplex and its elementary chain.

Consider the oriented n-simplex $\sigma = [x_0, \ldots, x_n]$ in which $x_i \in \mathbb{R}^N, i \in [0, \ldots, n]$ are its points. We define the *face operator* as

$$d_i(\sigma) := [x_0, \ldots, x_{i-1}, x_{i+1}, \ldots, x_n], \tag{2}$$

which returns the i^{th} $(n-1)$-simplex that is a *proper face* of the simplex σ. It follows that the *boundary operator* can be defined as

$$\partial_n(\sigma) := \sum_{i=0}^{n} (-1)^i d_i(\sigma). \tag{3}$$

Next consider a pair of simplices, $\sigma^p \in K$ and $\sigma^{p-1} \in K$, with $p \in [1, \ldots, n]$. If $\sigma^p \cap \sigma^{p-1} = \sigma^{p-1}$, then

let x_i be the extra point of σ^p that does not belong to σ^{p-1}. The *incidence number* is defined as

$$[\sigma^p, \sigma^{p-1}] := \begin{cases} 0, & \text{if } \sigma^p \cap \sigma^{p-1} = \emptyset; \\ 1, & \text{if } \sigma^{p-1} = d_i(\sigma^p); \\ -1, & \text{if } \sigma^{p-1} = -d_i(\sigma^p). \end{cases} \tag{4}$$

Let $f : K \mapsto \mathbb{N}$ be an injective map that assigns to each simplex in the mesh a non-negative integer. Then let

$$\begin{aligned} V &= \{\, v \mid v = f(\sigma) \in \mathbb{N}\,, \sigma \in K \,\}, \\ E &= \{\, e \mid e = (u, v), u = f(\sigma^p) \in V,\, v = f(\sigma^{p-1}) \in V,\, \sigma^p \in K,\, \sigma^{p-1} \in K,\, [\sigma^p, \sigma^{p-1}] \neq 0\} \end{aligned} \tag{5}$$

where $E \subset [V]^2$ is the set of integer pairs for which the corresponding simplices have a non-zero incidence number. The graph $G = (V, E)$ is used to represent the mesh simplicial complex, where V is the vertex set and E is the edge set. The map $f : K \mapsto V$ between the simplicial complex and the vertex set is bijective. Note that f is a *labeling* that has been extended not only to the points but to all simplices in K. It follows then that vertices represent simplices and edges represent their adjacency, and that by the definition of the incidence number, edges connect vertices that represent a simplex and its proper faces. The edges are assumed to be directed, i.e. if $e \in E$, $u = \text{tail}(e)$, $v = \text{head}(e)$ and $\sigma = f^{-1}(u)$, $\tau = f^{-1}(v)$, then $\dim(\sigma) = \dim(\tau) + 1$, which renders the graph *directed, oriented* (i.e. no loops or multiple edges) and *acyclic*.

The *in-degree* of a vertex v is the number of edge heads adjacent to it and is denoted as $d^-(v)$. Conversely, the *out-degree* is the number of edge tails adjacent to v, denoted as $d^+(v)$. The *source vertex set* of a vertex is defined as $D^-(v) = \{u \mid u = \text{tail}(e)\, \forall e \text{ s.t. } v = \text{head}(e)\}$, and the *target vertex set* of a vertex is defined as $D^+(v) = \{u \mid u = \text{head}(e)\, \forall e \text{ s.t. } v = \text{tail}(e)\}$.

A *path* on the graph G is defined as the sequence $P = v_1 e_1 \dots v_i e_i \dots e_{n-1} v_n$ where both the vertices $\{v_1, \dots, v_n\} \subset V$ and the edges $\{e_1, \dots, e_{n-1}\} \subset E$ are distinct. The tail and head vertices of edge e_i are v_i and v_{i+1}, correspondingly, with the initial and terminal vertices of P being $v_1 = \text{init}(P)$ and $v_n = \text{ter}(P)$, respectively. The length of the path from vertex u to vertex v is the number of edges in the path, and a path of length k is denoted as P^k. The distance $d(u, v)$ is the length of the shortest path between the vertices. Note that for the graph G constructed as described above, the length and the distance are equal. Furthermore, if there exists a path between two vertices, their distance is given by

$$d(u, v) = |\dim(\sigma) - \dim(\tau)|, \quad \sigma = f^{-1}(u), \quad \tau = f^{-1}(v). \tag{6}$$

Next consider the simplex σ^p. All paths of length p with initial vertex $f(\sigma^p)$ have terminal vertices that correspond to the points that define the simplex, i.e.

$$\sigma^p = [f^{-1}(v_0), \dots, f^{-1}(v_i), \dots, f^{-1}(v_p)], \quad v_i \in \{\, v \mid v = \text{ter}(P^p)\, \forall\, \text{init}(P^p) = f(\sigma^p) \,\}. \tag{7}$$

Now let $\mathcal{K}$ be the collection of all point subsets $\{x_0, \dots, x_p\}$ such that each subset spans a simplex in K. The collection $\mathcal{K}$ is known as the *vertex scheme* of K and is a prime example of an *abstract simplicial complex* [19]. Define $\mathcal{S}$ as the collection of sets $a^p = \{v_0, \dots, v_i, \dots, v_p\}$ with v_i given by Eq. (7) for each of the simplices $\sigma^p \in K$. The collection $\mathcal{S}$ is also an abstract simplicial complex in which the sets $a^p \in \mathcal{S}$ are its simplices. Thus, there is a bijective correspondence f mapping the vertex set of $\mathcal{K}$ to the vertex set of $\mathcal{S}$, and therefore $\mathcal{K}$ and $\mathcal{S}$ are isomorphic [19]. By virtue of this isomorphism, our graph representation is *complete* in the sense of Weiler [30]. The simplicial complex K is a *geometric realization* of the abstract

simplicial complex $\mathcal{S}$. Furthermore, the abstract simplicial complex $\mathcal{S}$ is a *partially ordered set* in which the *covering relation* is defined as $a^{p-1} \subset a^p$. Thus, the graph G is in effect a *Hasse diagram* of the abstract simplicial complex $\mathcal{S}$ where the representation of the empty set has been omitted.

In addition, the graphs obtained by representing a simplicial complex in this manner are $(n+1)$-partite (i.e. they have $n+1$ "levels", from points to the simplex of highest dimension), and are also simplicial complexes themselves [2, 13].

Let $g : V \mapsto S \subset \mathbb{N} \mid e = (u, v) \in E,\ g(u) \neq g(v)$ be a surjective map that assigns to each vertex in the graph an integer that is different from the integer corresponding to the vertices adjacent to it. This map is a *vertex coloring* of the graph. The cardinality of the smallest set S is the *vertex chromatic number* $\chi(G)$. The graph G has $\chi(G) = 2$; nevertheless, for ease of visualization and convenience in implementation, we choose the vertex coloring $g : V \mapsto S \mid g(f(\sigma)) = \dim(\sigma)\, \forall\, \sigma \in K$ and therefore $S = \{0, \ldots, n\}$.

Let $h : E \mapsto S$ be a surjective map that assigns to each edge in the graph an integer in S. We choose the map $h : E \mapsto S \mid h(e) = i\,, e \in E,\ f^{-1}(\text{head}(e)) \equiv +d_i(f^{-1}(\text{tail}(e)))$. Note that the ordering of the proper faces is important here. Although this map is not an *edge coloring* from the point of view of graph theory (viz. there are adjacent edges that map to the same integer), we associate the integers in S with colors (0 = red, 1 = green, 2 = blue, ...), which not only enhances the visualization of graphs, but also aids in the determination of simplex orientation, as described later.

The 0-simplex or point has the simplest representation as a graph that consists of a single, isolated vertex shown in Fig. 1 as a red oval.

Figure 1: 0-simplex or point.

The use of interpolation functions of order higher than linear requires additional points that are not present in a strictly simplicial mesh. These additional points are easily accommodated by simply including them, and play no defining role in the topological manipulations of the mesh. Thus, although an n-simplex has $n+1$ 0-simplices or points [17, 19, 20], the graph representation of a simplex may include additional points to accommodate higher-order finite elements.

In order to illustrate the graph representation for higher order elements, henceforth we consider the particular case of quadratic interpolation functions. Thus, the 1-simplex or segment is augmented with an extra point as shown in Fig. 2. Both segments are represented by green vertices with their proper faces, points 0 and 1, connected to them by colored, directed edges. Both segments have been augmented by midpoint 2 to accommodate quadratic interpolation functions.

The segments in Fig. 2 have opposite orientations. This is represented in the graph by edge colors. Disregarding the midpoint, the oriented segment in Fig. 2a is represented as $[x_0, x_1]$, whereas the bottom segment is $[x_1, x_0]$. Thus, the ordering of the points differs by an odd permutation.

The graph representation of the 2-simplex or triangle is shown in Fig. 3. Both triangles in the figure are represented as $[x_0, x_1, x_2]$ since the arrangement of the corner points differs only by an even permutation [19, §5]. Note, however, that the difference in ordering is recorded in the graph as an even permutation of the colors of the edges connecting the triangles with their segments.

A finite element mesh in the context of the graph representation is a simplicial complex where an extra graph vertex is introduced, as shown in Fig. 4. This *root* vertex, shown as a white oval in the figure, guarantees that the graph is *weakly connected*, regardless of whether the mesh is intact or in a highly fragmented

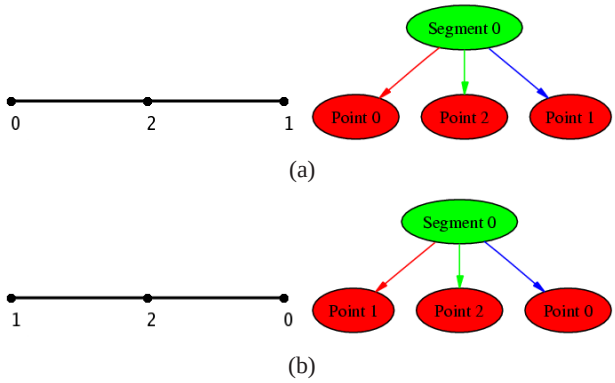


Figure 2: 1-simplex or segment.

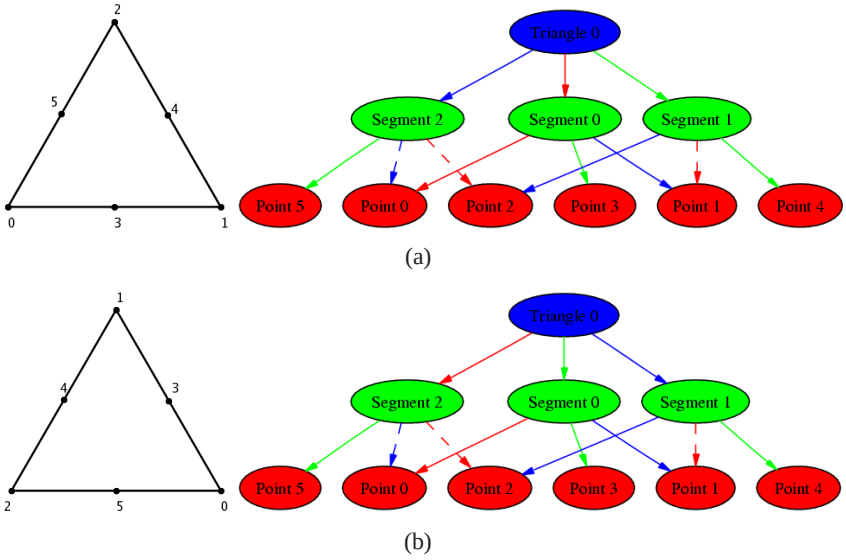


Figure 3: 2-simplex or triangle.

state. This greatly simplifies the manipulation of the graph. For the tetrahedron in Fig. 4, the colors of the edges define an ordering of the vertices that is considered canonical, and any deviations from it are recorded within the graph, as described next.

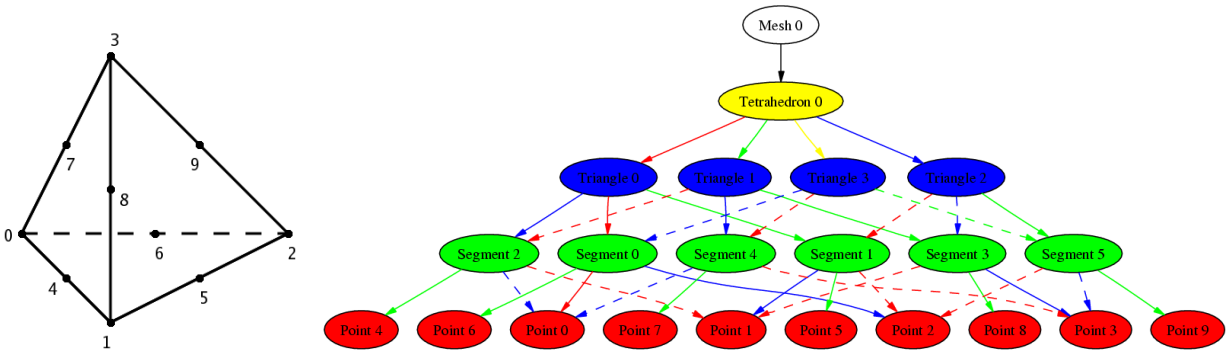


Figure 4: Single-tetrahedron mesh and its corresponding graph.

3 Ordering and Orientation

Within the context of algebraic topology, all even permutations of a selected ordering of vertices represent the same orientation of a simplex. Nevertheless, in the case of frequent interaction with traditional finite element connectivity arrays (e.g., using the graph representation within an existing finite element code), it is necessary to record additional orientation information in the graph in order to fully recover these arrays.

The added complexity is required to preserve the order of the nodal connectivity when exchanging information between the graph and connectivity arrays. A nodal connectivity array is essentially a collection of memory address offsets, and therefore it is crucial to preserve its ordering to avoid accessing incorrect or invalid memory locations. Thus, the graph representation is endowed with the property that any alteration of the ordering of the points is recorded, regardless of the reordering being an even or odd permutation, as shown in Fig. 3.

Consider now the case of two triangles that share a segment, as shown in Fig. 5. The order of segment 0 changes depending on whether it is seen as a proper face of triangle 0 or triangle 1 (i.e., segment $0 = +d_0(\text{triangle } 0) = -d_1(\text{triangle } 1)$). This is reflected in Fig. 5a, where the edge joining triangle 0 with segment 0 is red, whereas the edge joining triangle 1 with the same segment is green. Additionally, the orientation of the segment is also different according to whether it is considered part of one triangle or the other, and the corresponding graphs record this difference as a swap of the colors of the edges that join the segment with points 0 and 1.

When both triangles form part of the same simplicial complex, however, the graph representation contains a single instance of the shared segment, as shown in Fig. 5b. Therefore, each edge joining the segment with a point records a single color, which by convention is the color assigned by the first simplex that references the edge. In order to avoid the loss of ordering information of points with respect to the segment for each of the original triangles, edge color maps are introduced.

An edge color map is a simple bijective transformation that records $(n-1)$-simplex ordering information for an n-simplex when the latter is a shared face of several $(n+1)$-simplices. Let $e \in E$ be an edge in the graph. Then let $u = \text{tail}(e)$, $v = \text{head}(e)$, $\sigma = f^{-1}(u)$, and $\tau = f^{-1}(v)$, with $\dim(\tau) > 0$. Then it

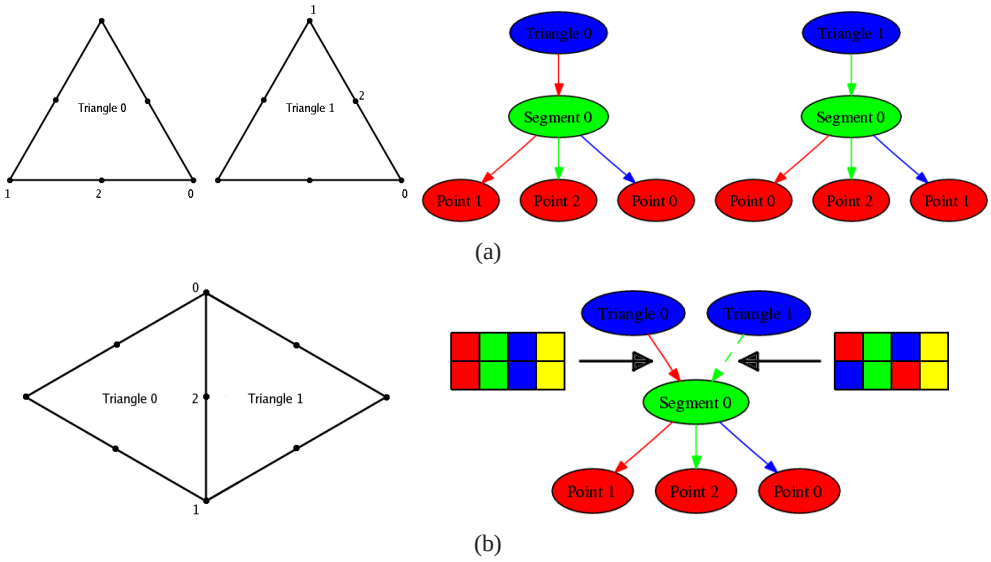


Figure 5: Local Ordering as Color Maps.

follows that $d^+(v) = \dim(\tau) + 1$ and that the edge color map can be defined as $C : \{0, \dots, \dim(\tau)\} \mapsto \{0, \dots, \dim(\tau)\} \mid d_{C(g(\hat{e}))}(d_{g(e)}(\sigma)) = d_{g(\hat{e})}(\tau)\,, \forall\, \hat{e}\,\text{s.t.}\, v = \text{tail}(\hat{e})$. Edges are thus endowed with a color map property, besides their own color.

The color maps in Fig. 5b indicate the color translation that must be applied to the edges joining the segment and points to recover the ordering shown in Fig. 5a. Thus, the color map for the edge joining triangle 0 and segment 0 is the identity map, which indicates that no color transformation is necessary to recover the original colors of the edges joining the segment with its points. By contrast, the color map associated with the edge joining triangle 1 and segment 0 indicates that to recover the color of the edges joining the segment with its points shown in Fig. 5a for triangle 1, blue must be replaced by red and red by blue, while green is left intact (the fourth color, yellow, is included to accommodate tetrahedra, thus allowing for the use of constant-sized color maps for all edges, which greatly simplifies the implementation). The fact that the color map for the edge joining triangle 1 and segment 0 in Fig. 5b is not the identity is indicated by the dashed line used to represent the edge.

4 Graph Fracture

Let c_p be the p-chain in K such that $c_p = \sum \sigma^p \,\forall\, \sigma^p \in K$. We define the chain of internal interfaces as $c_{I,n-1} := c_{n-1} - \partial_n(c_n)$. On the graph G, the set of vertices corresponding to the internal interfaces is given by $V_{I,n-1} = \{v \mid v \in V,\, d^-(v) = 2,\, d^+(v) = n\}$, and by analogy to c_p we define $V_p = \{v \mid v \in V,\, g(v) = p\}$.

The first step in the graph fracture procedure consists of identifying open simplices. We define an open simplex as one that is adjacent to a simplex in $c_{I,n-1}$ that has been identified as fractured by a failure criterion. Hence, in a three-dimensional simulation, if a triangle (2-simplex) is marked as open, then all its faces (segments and points) and the adjacent tetrahedra are marked as open too. On the graph G, the corresponding vertices in $V_{I,n-1}$ and all vertices in all paths that start or end in an open vertex in $V_{I,n-1}$ are

considered open as well, and the set that contains them is denoted as V_F.

The *star* Stσ of a simplex $\sigma \in K$ is defined as the union of the interiors of the simplices in K that have σ as a face. The star of a simplex corresponds to a subgraph G' of G, with V' and E' its corresponding vertex and edge sets. Moreover, G' is an *induced* subgraph of G, in the sense that if two vertices u and v in V' are joined by an edge in E', then there is a corresponding edge in E that also joins u and v, i.e, $e' = (u, v) \in E',\ u, v \in V' \implies e = (u, v) \in E,\ u, v \in V$. We call the subgraph G' corresponding to the star Stσ the simplex subgraph of $v = f(\sigma)$ and denote it as $G'(v)$. The number of (weakly) *connected components* of a graph G is $N(G)$ and the components themselves are $G_1, \ldots, G_{N(G)}$. An *articulation point* of a graph G is a vertex v such that $G \setminus v$ has more connected components than G.

Let $V_{F,i} = \{v \mid v \in V_F,\ g(v) = i\}$ be the set of open vertices that correspond to open i-simplices, with $i \in \{0, \ldots, n\}$. The fracture algorithm is effected by performing SPLIT$(G, V_{F,0}, n, 0)$, with the SPLIT procedure defined in Algorithm 1.

Algorithm 1 SPLIT(G, U, n, i) Split articulation points.

Require: $U \subset V_i,\ i \leq n - 1$

for all $v \in U$ **do**
  if $i < n - 2$ **then**
    SPLIT$(G, D^-(v), n, i + 1)$
  else
    CLONE$(G, D^-(v), n)$
  end if
  $G'' \leftarrow G'(v) \setminus v$ // Check whether v is an articulation point
  for all $j \in \{2, \ldots, N(G'')\}$ **do**
    $Y \leftarrow \{u \mid u \in G''_j,\ g(u) = i + 1\}$
    $V' \leftarrow \{V', z\}$ // Split the vertex in the subgraph and graph
    for all $u \in Y$ **do**
      $E' \leftarrow E' \setminus (u, v)$
      $E' \leftarrow \{E', (u, z)\}$
    end for
  end for
end for

Note that the fracture algorithm is defined recursively, and therefore its expression is relatively simple. The isomorphism between the changing graph G and simplicial complex K is preserved throughout these operations. The cloning operation only duplicates internal interface simplices, and thus preserves the isomorphism between G and K. The splitting operation applies only to articulation points (or articulation simplices in K) that join otherwise independent components of the mesh. The corresponding vertex and simplex are split according to the number of components, and therefore the isomorphism is also preserved.

A sequence of operations comprising the fracture algorithm is further illustrated in Fig. 6 for a test fracture problem involving two tetrahedra sharing a face. At the outset, this common face is assumed to have undergone fracture and then the algorithm proceeds, as required, to completely separate the two tetrahedra.

An important feature of the graph representation is its ability to correctly handle non-manifold meshes. In order to further elucidate this point, we employ the fracture algorithm to open all internal faces of the two test meshes shown in Fig. 7. The first mesh consist of two identical cubes with a common vertex and the second mesh of the same two cubes but now sharing an edge. The graph-based fracture algorithm is clearly capable of correctly handling the two non-manifold meshes, as is evident from Fig. 7. For the purpose of

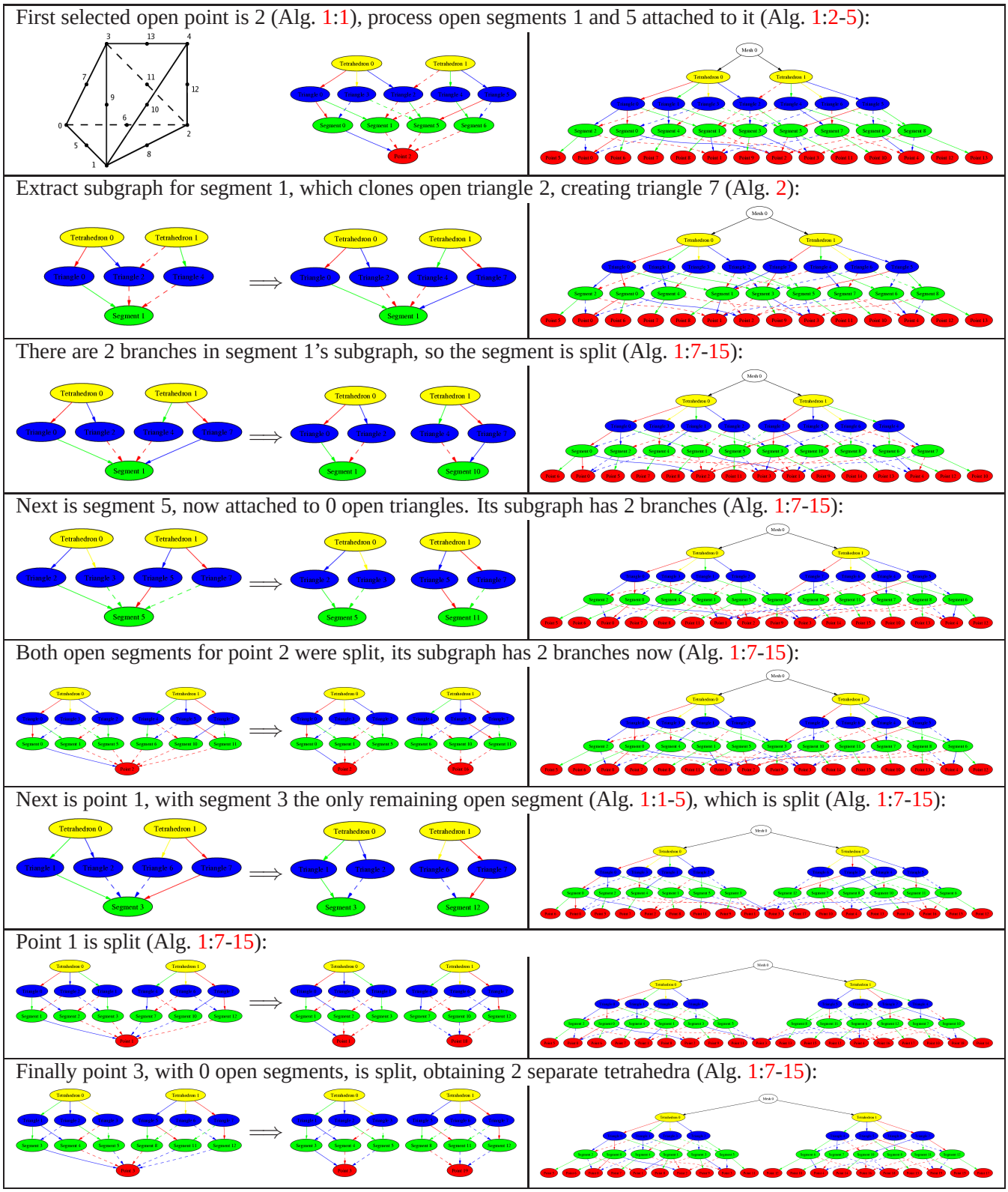
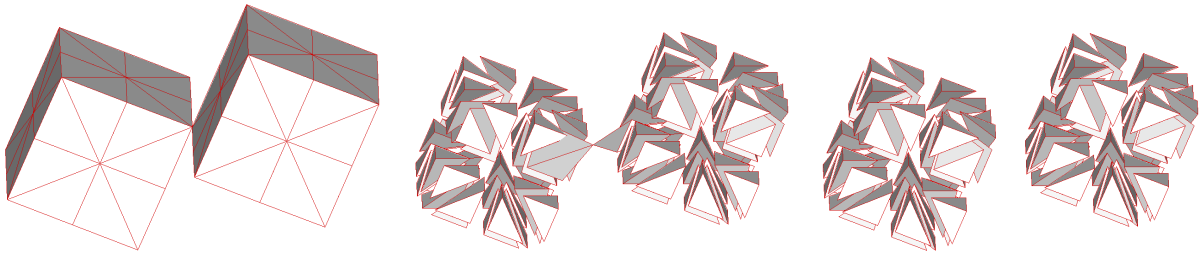


Figure 6: Fracture procedure for two joined tetrahedra.

Algorithm 2 CLONE(G, U, n) Duplicate fractured interface simplices.

Require: $U \subset V_{I,n-1}$
for all $v \in U$ **do**
  if $v \in V_{F,n-1}$ **then**
    $Y \leftarrow D^-(v)$ // Note that $d^-(v) = 2$, hence $|Y| = 2$
    $u_1 \leftarrow u \in Y$ s.t. $[f^{-1}(u_1), f^{-1}(v)] = 1$
    $u_2 \leftarrow u \in Y$ s.t. $[f^{-1}(u_2), f^{-1}(v)] = -1$
    $V \leftarrow \{V, w\}$
    $E \leftarrow E \setminus (u_2, v)$
    $E \leftarrow \{E, (u_2, w)\}$
    for all $z \in D^+(v)$ **do**
      $E \leftarrow \{E, (w, z)\}$
    end for
  end if
end for

comparison, we also show the results obtained by the application of the classical fracture algorithm proposed by Pandolfi and Ortiz [22, 23], which does not yield the expected outcome.

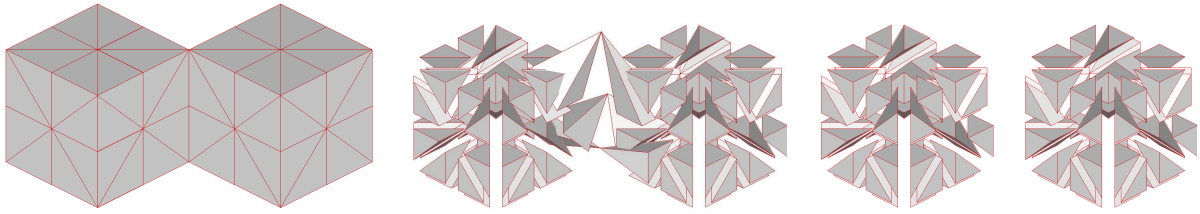


(a) Original mesh (b) Fracture from [22] (c) Graph fracture

Figure 7: Two cubes joined by a point.

Next, we investigate the performance of the graph representation for problems involving fracture and fragmentation of tetrahedral finite element meshes. Our implementation of the graph representation has been developed using the Boost Graph Library (BGL) [26]. The BGL applies the principles of generic programming for the construction of advanced data structures and algorithms commonly used in graph theory. The use of the BGL helps to significantly reduce the development time and offers excellent flexibility in the handling of various data types and related algorithms.

We compare the performance of our BGL-based implementation to the reference fracture algorithm of Pandolfi and Ortiz [22, 23]. To this end, we evaluate the computational cost of performing two distinct operations: the construction of the initial representation of a tetrahedral mesh and the separation of all tetrahedra in the mesh by way of fracture. In Fig. 9, we plot the time in seconds necessary to build the initial representation of a mesh as a function of the number of elements for twenty different tetrahedral meshes. The green color in the plot corresponds to the graph representation, whereas the red color to the reference implementation, respectively. Clearly, the graph representation exhibits linear dependence of the initialization time with respect to the size of the finite element mesh. By contrast, the initialization time for the reference



(a) Original mesh (b) Fracture from [22] (c) Graph fracture

Figure 8: Two cubes joined by an edge.

implementation is essentially quadratic. The massive savings offered by the graph representation become fully realized for meshes composed of approximately 200,000 elements: the initialization time is reduced from days to mere minutes.

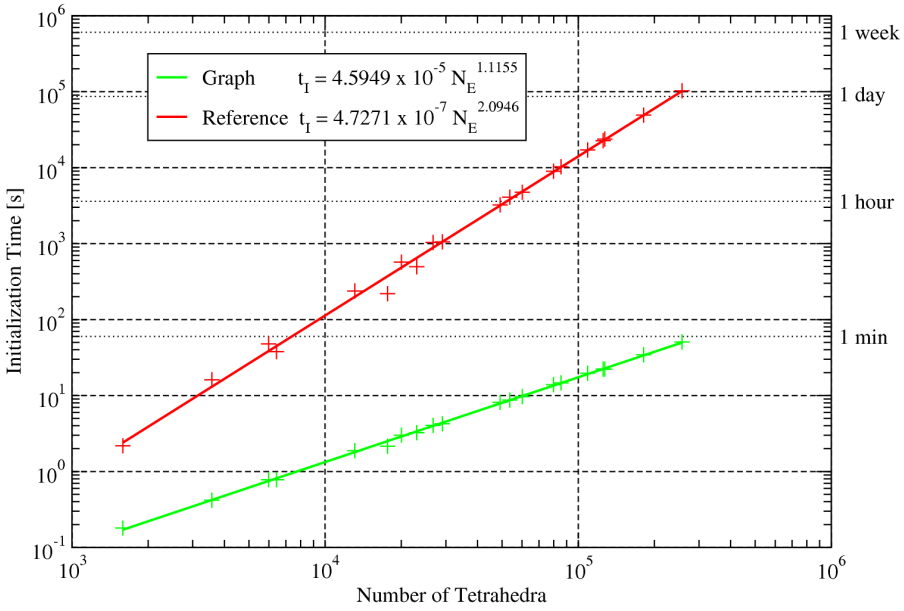


Figure 9: Initialization time. Comparison of 20 meshes.

Fig. 10 contains a plot of the time required to fracture all internal interfaces in a finite element mesh as a function of the number of such interfaces in the mesh. The plot was obtained using the same set of meshes as for the initialization study. Again the green color in the plot corresponds to the graph representation, and the red color to the reference implementation. The Pandolfi-Ortiz algorithm provides better performance for relatively small meshes, i.e. containing less than approximately 80,000 internal interfaces (corresponding to approximately the same number of elements). In the case of larger meshes, however, an opposite trend is observed as the graph-based fracture algorithm furnishes substantial performance benefits. More importantly, the cost of the graph-based algorithm appears to be merely linear in the number of interfaces undergoing fracture, whereas it is nearly quadratic for the Pandolfi-Ortiz algorithm.

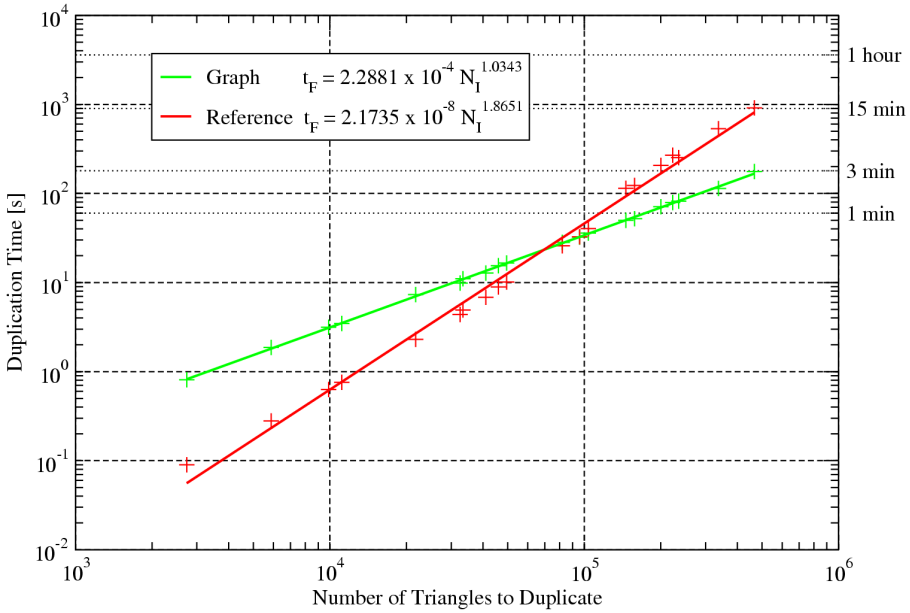


Figure 10: Fracture time. Comparison of 20 meshes.

5 Numerical example

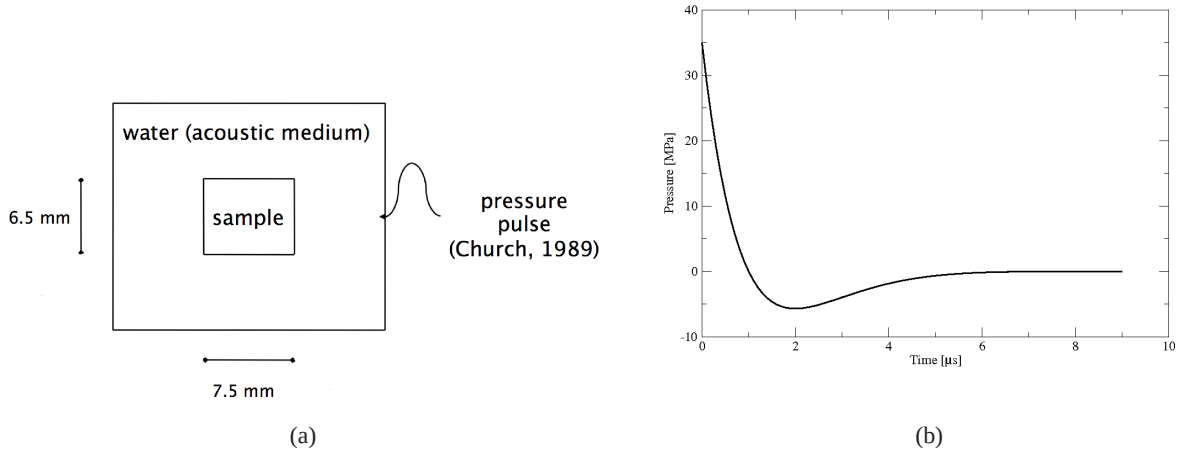
In order to demonstrate the power of the graph-based fracture algorithm, we perform a computer simulation of the fracture and fragmentation of artificial kidney stones. Lithotripsy is a medical procedure designed to induce stresses in renal calculi (kidney stones) through the repeated application of pressure pulses to the exterior of the human body. The goal of the procedure is to break up the kidney stones into small fragments that, subsequently, can exit the kidney through the urethra. The pressure pulses are commonly generated in a lithotripter. The ability of the lithotripter to fragment kidney stones is customarily investigated experimentally. These experiments, however, employ gypsum cylinders in lieu of renal calculi. The cylinders are submerged in water as to simulate the effect of live tissue. A series of identical pressure pulses is applied to the cylinders in the form of a planar front along the cylinder axis. The appearance of cracks in the cylinders depends on both the peak pressure of the pulse and the number of applied pulses. Moreover, a salient feature of the crack pattern is formation of a spall plane at 2/3 of the cylinder length.

5.1 Gypsum experiment

Shock wave lithotripsy has become the primary technique in the treatment of renal calculi. However, much controversy still remains as to precise mechanisms leading to the comminution of kidney stones. At present, brittle fracture is widely believed to be the main mechanism leading to fragmentation [15]. In view of the difficulties associated with accurate control and instrumentation of the kidney stone fracture in live tissue, but also to study lithotripter performance separately, artificial stones are generally used in the research of the fragmentation mechanisms. The Ultracal-30 gypsum is most frequently adopted as a model for renal calculi, as it is suitable for lithotripsy studies in vitro, acute animal experiments in which the stones are implanted in the kidney, and as a target to compare in vitro performance of intra-corporeal lithotripters [18].

A typical experimental setup involving the Ultracal-30 gypsum cylinders is shown in Fig. 11a [18]. The

cylinders are loaded by a pressure pulse along the cylinder axis. The pressure profile of the pulse is plotted in Fig. 11b.



(a) (b)

Figure 11: Gypsum stone experiment setup (a) with pressure pulse applied to the gypsum sample (b).

Recovered gypsum cylinders are presented in Fig. 12. The formation of a spall plane at 2/3 of the cylinder length is clearly visible. Also, the three-dimensional surface reconstruction with damage signs due to cavitation is shown.

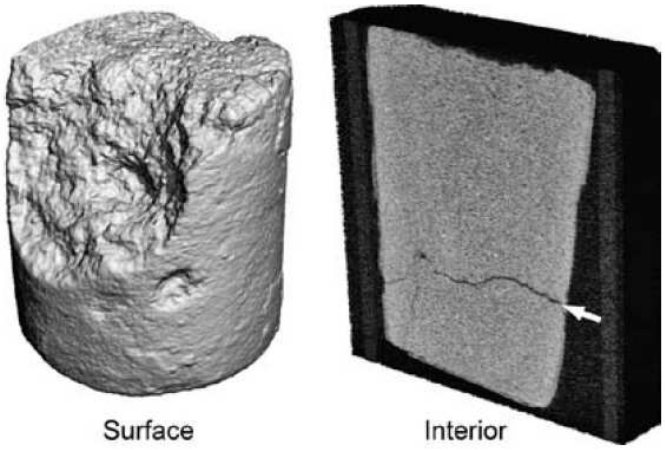
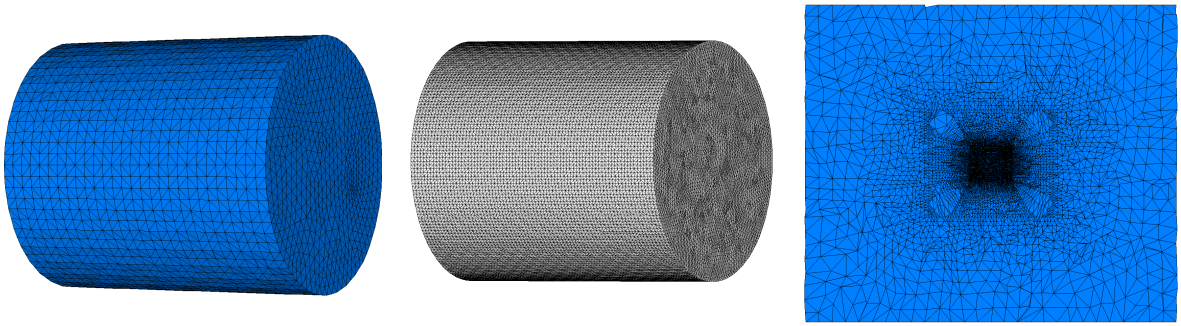


Figure 12: 3D surface reconstruction (left) shows damage from cavitation, and cross-section (right) shows the characteristic spallation plane [18].

5.2 Finite element simulation

In our finite-element simulation of the artificial kidney stone lithotripsy we explicitly model both the gypsum cylinder and the surrounding water. A very fine discretization of the cylinder is introduced in order to provide a suitable number of possible fracture paths. Overall, the mesh, which is shown in Fig. 13, includes 223,020 tetrahedral elements (130,202 for gypsum and 92,818 for water).



(a) (b) (c)

Figure 13: Finite element mesh used in the simulation. (a) Mesh for water. (b) Mesh for gypsum cylinder. (c) Cross section showing the finer gypsum mesh embedded in the coarser water mesh.

The mechanical response of water is assumed to be well represented by an acoustic element that allows for the propagation of pressure waves only. The finite-deformation acoustic formulation is displacement-based in order to simplify the solid–fluid interaction. Gypsum is modeled as a neo-hookean finite-deformation elastic solid. In the simulation, we use composite tetrahedral finite elements [27].

The material properties employed in the simulation are shown in Table 1, where E, ν, ρ_g, σ_c and G_c denote the Young's modulus, Poisson's ratio, mass density, fracture critical stress and fracture energy of the gypsum, respectively, and K and ρ_w are the bulk modulus and mass density of water.

E [GPa]	ν	ρ_g [kg m^{-3}]	σ_c [MPa]	G_c [J m^{-2}]	β^2	K [GPa]	ρ_w [kg m^{-3}]
10.8	0.3461	1700	6.0	21.88	5.0	2.25	1000

Table 1: Material parameters used in the simulation

We present the results of our finite-element simulation in Fig. 14. The sub-figures show different perspectives of the final state of the gypsum sample, with cracks represented by a solid color and the sample itself as a translucent cylinder. The color corresponds to the component of the normal stress along the axis of the cylinder, with blue for lower values and green for higher values. The simulation is capable of capturing all of the essential components of the experiments. Thus, when the pressure wave enters the cylinder, a network of frontal cracks develops (Fig. 14a). As the wave proceeds along the axis of the cylinder, a failure cone forms, a typical phenomenon associated with fracture under compression (Figs. 14b, 14c, 14d). The formation of a spall plane at 2/3 length may be observed in all sub-figures in Fig. 14, in particular in Fig. 14c. Finally, exfoliation is best observed in Fig. 14e. The salient feature of these experiments, the formation of the spall plane, is evidently captured very well in our simulation.

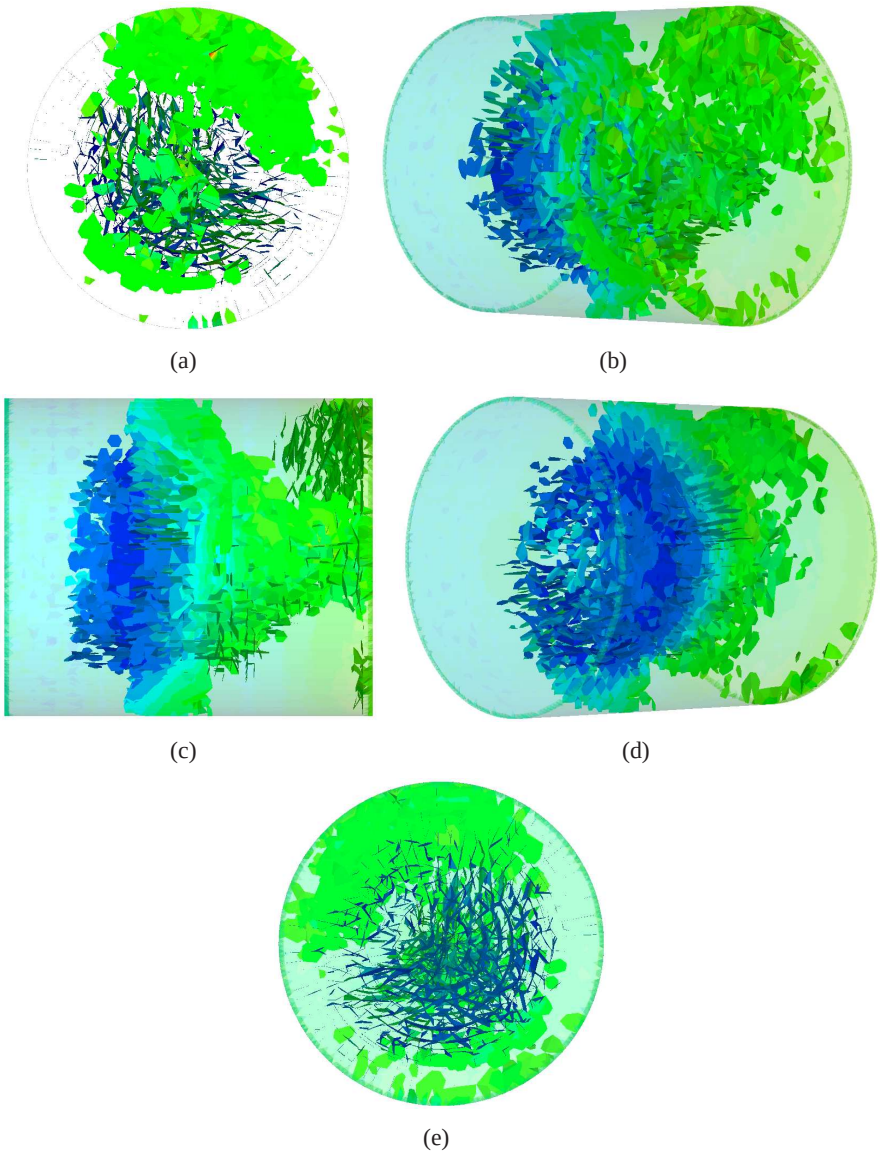


Figure 14: Gypsum fracture simulation results: (a) Frontal cracks. (b) Surface and internal cracks. (c) Spall at 2/3 of length. (d) Spall and failure cone. (e) Exfoliation and spall.

6 Conclusions

We have developed a graph representation of simplicial finite element meshes that is isomorphic to the original simplicial complex, and therefore preserves all topological information. By extending the graph representation to include ordering information, legacy finite element codes can make use of the graph representation for complex topological operations, thus minimizing the changes needed in the legacy code. In particular, the graph representation is well-suited for fracture, as demonstrated by the simulation of the brittle fracture of an artificial kidney stone. The required operations for fracture in the graph result in simple recursive algorithms capable of handling non-manifold topologies. In addition, the initialization performance of the graph fracture is superior to the reference implementation and better suited for fracture for large meshes. Furthermore, the localized nature of the operations needed for fracture render the graph representation specially well-suited for parallel implementation. The performance of the parallel implementation of the method deserves detailed analysis and will be addressed in a subsequent publication.

Acknowledgements

We gratefully acknowledge the support of the Department of Energy through Caltech's ASC Center for the Simulation of the Dynamic Response of Materials. This work was performed in part under the auspices of the U.S. Department of Energy by the University of California, Lawrence Livermore National Laboratory under Contract W-7405-Eng-48.

References

[1] I. Babuska and J.M. Melenk. The partition of unity method. *Internat. J. Numer. Meths. Engrg.*, 40: 727–758, 1997.

[2] J. Bang-Jensen and G. Gutin. *Digraphs: Theory, Algorithms and Applications*. Springer-Verlag, London, 2002.

[3] M.W. Beall and M.S. Shephard. A general topology-based mesh data structure. *Internat. J. Numer. Meths. Engrg.*, 40:1573–1596, 1997.

[4] T. Belytschko and T. Black. Elastic crack growth in finite elements with minimal remeshing. *Internat. J. Numer. Meths. Engrg.*, 45:601–620, 1999.

[5] T. Belytschko, Y.Y. Lu, and L. Gu. Element-free galerkin methods. *Internat. J. Numer. Meths. Engrg.*, 37:229–256, 1994.

[6] G.T. Camacho and M. Ortiz. Computational modeling of impact damage in brittle materials. *Internat. J. Solids and Structures*, 33:2899–2938, 1996.

[7] W. Celes, G. Paulino, and R. Espinha. A compact adjacency-based topological data structure for finite element mesh representation. *Internat. J. Numer. Meths. Engrg.*, 2005. In print.

[8] J. Q. Clayton and J. F. Knott. Observations of fibrous fracture modes in a prestrained low-alloy steel. *Metal Science*, 10:63–71, 1976.

[9] C. Daux, N. Moes, J. Dolbow, and T. Belytschko. Arbitrary branched and intersecting cracks with the extended finite element method. *Internat. J. Numer. Meths. Engrg.*, 48:1741–1760, 2000.

[10] R. de Borst, J.J.C Remmers, A. Needleman, and M.A. Abellan. Discrete vs smeared crack models for concrete fracture: bridging the gap. *Int. J. Numer. Anal. Meth. Geomech.*, 28:583–607, 2004.

[11] S. Deo. *Algebraic Topology: A Primer*. Hindustan Book Agency, New Delhi, 2003.

[12] R.V. Garimella. Mesh data structure selection for mesh generation and FEA applications. *Internat. J. Numer. Meths. Engrg.*, 55(4):451–478, 2002.

[13] J.L. Gross and T.W. Tucker. *Topological Graph Theory*. Dover, Mineola, NY, 2001.

[14] A.R Ingraffea and V. Saouma. Numerical modelling of discrete crack propagation in reinforced and plain concrete. In *Fracture Mechanics of Concrete*, pages 171–225. Martinus Nijhoff Publishers, Dordrecht, 1985.

[15] M. Lokhandwalla and B. Sturtevant. Fracture mechanics model of stone comminution in eswl and implications for tissue damage. *Physics in Medicine and Biology*, 45:1923–1940, 2000.

[16] J. J. Mason, A. J. Rosakis, and G. Ravichandran. On the strain and strain-rate dependence of plastic work converted to heat-an experimental study using high-speed infrared detectors and the kolsky bar. *Mechanics of Materials*, 17:135–145, 1994.

[17] C.R.F. Maunder. *Algebraic Topology*. Dover, Mineola, NY, 1980.

[18] J.A. McAteer, J.C. Williams, R.O. Cleveland, J van Cauwelaert, M.R. Bailey, D.A. Lifshitz, and A.P. Evan. Ultracal-30 gypsum artificial stones for research on the mechanisms of stone breakage in shock wave lithotripsy. *Urological Research*, 33(6):429–434, 2005.

[19] J.R. Munkres. *Elements of Algebraic Topology*. Perseus Publishing, Cambridge, MA, 1984.

[20] J.R. Munkres. *Topology*. Prentice Hall, Upper Saddle River, NJ, 2000.

[21] M. Ortiz and A. Pandolfi. Finite-deformation irreversible cohesive elements for three-dimensional crack-propagation analysis. *Internat. J. Numer. Meths. Engrg.*, 44:1267–1282, 1999.

[22] A. Pandolfi and M. Ortiz. Solid modeling aspects of three-dimensional fragmentation. *Eng. Comput.*, 14(4):287–308, 1998.

[23] A. Pandolfi and M. Ortiz. An efficient adaptive procedure for three-dimensional fragmentation simulations. *Eng. Comput.*, 18(2):148–159, 2002.

[24] J.F. Remacle and M.S. Shephard. An algorithm oriented mesh database. *Internat. J. Numer. Meths. Engrg.*, 58(2):349–374, 2003.

[25] H. C. Rogers. Adiabatic plastic deformation. *Annual Review of Materials Science*, 9:283–311, 1979.

[26] J.G. Siek, L.Q. Lee, and A. Lumsdaine. *The boost graph library: user guide and reference manual*. Addison-Wesley Longman Publishing Co., Inc., Boston, MA, USA, 2002.

[27] P Thoutireddy, J.F. Molinari, E.A. Repetto, and M. Ortiz. Tetrahedral composite finite elements. *Internat. J. Numer. Meths. Engrg.*, 53:1337–1351, 2002.

[28] S.P. Timothy. The structure of adiabatic shear bands in metals: A critical review. *Acta Metallurgica*, 35:301–306, 1987.

[29] V. Tvergaard. Material failure by void growth to coalescence. *Advances in Applied Mechanics*, 27: 83–151, 1990.

[30] K. Weiler. *The Radial Edge Structure: A Topological Representation for Non-Manifold Geometric Boundary Modeling*, pages 3–36. North-Holland, 1988.

[31] A. T. Winter. The influence of projectile geometry on adiabatic shear and target failure. *Metallurgical Transactions*, 4:1829–1833, 1973.

[32] Y.B. Xu, Z.G. Wang, X.L. Huang, D. Xing, and Y.L. Bai. Microstructure of shear localization in low carbon ferrite-pearlite steel. *Materials Science and Engineering*, A114:81–87, 1989.

[33] Q. Yang, A. Mota, and M. Ortiz. A class of variational strain-localization finite elements. *Internat. J. Numer. Meths. Engrg.*, 62:1013–1037, 2004.

[34] M. Zhou, A.J. Rosakis, and G. Ravichandran. Dynamically propagating shear bands in impact-loaded prenotched plates .1. experimental investigations of temperature signatures and propagation speed. *Journal of the Mechanics and Physics of Solids*, 44(6):981–1006, 1996.

[35] M. Zhou, A.J. Rosakis, and G. Ravichandran. Dynamically propagating shear bands in impact-loaded prenotched plates .2. numerical simulations. *Journal of the Mechanics and Physics of Solids*, 44(6): 1007–1025, 1996.